Matrix normal scale_r_cor_by_cov brms function

Jordan Scott Martin

7/29/2021

```r
########################################################################
#generate Stan code for brms model with Kronecker product using by function
###########################################################################

#simulate scaling matrix
A = rethinking::rlkjcorr(1, 100, 1)

#GROUP A
#simulate correlated phylogenetic effects
r_G = -0.5 #phylo correlation
v_G =  0.5 #phylo variance

G_cor1 <- matrix(c(1,r_G,r_G,1), nrow=2, ncol=2)
G_sd1  <- c(sqrt(v_G),sqrt(v_G))
G1 <- diag(G_sd1) %*% G_cor1 %*% diag(G_sd1)

r_G =  0.5 #phylo correlation
v_G =  0.5 #phylo variance

G_cor2 <- matrix(c(1,r_G,r_G,1), nrow=2, ncol=2)
G_sd2  <- c(sqrt(v_G),sqrt(v_G))
G2 <- diag(G_sd2) %*% G_cor2 %*% diag(G_sd2)

Gblock = as.matrix(Matrix::bdiag(G1,G2))

Kron.prod1 <- Gblock %x% A
P <- matrix(mvtnorm::rmvnorm(1, mean=rep(0,nrow(A)*4), sigma=Kron.prod1), ncol = 4)
P1 = P[c(1:50),1:2];cor(P1)
P2 = P[c(51:100),3:4];cor(P2)

#group id
P = rbind(P1,P2)

#Gaussian responses
v_res = 0.5 #residual variance (assume independent errors)
t1 = 0 + P[,1] + rnorm(nrow(A), 0, sqrt(v_res))
t2 = 0 + P[,2] + rnorm(nrow(A), 0, sqrt(v_res))

library(brms); library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
```

```
df = data.frame(t1,t2, phylo = seq(1:nrow(A)),
                group = rep(c(1, 2), each = nrow(A)/2))

rownames(A) = seq(1:nrow(A))

write(
  make_stancode(formula = bf(t1 ~ 0 + (1|G| gr(phylo, cov = A, by = group))) +
                    bf(t2 ~ 0 + (1|G| gr(phylo, cov = A, by = group)) ),
                  data = df, data2 = list(A = A),
                  prior = c(prior("exponential(1)", class = "sd", resp = "t1"),
                            prior("exponential(1)", class = "sigma", resp = "t1"),
                            prior("exponential(1)", class = "sd", resp = "t2"),
                            prior("exponential(1)", class = "sigma", resp = "t2"),
                            prior("lkj(1)", class = "cor"))),
  "m_1.stan")

#get stan data list
stan_data = make_standata(formula = bf(t1 ~ 0 + (1|G| gr(phylo, cov = A, by = group))) +
                                bf(t2 ~ 0 + (1|G| gr(phylo, cov = A, by = group)) ),
                            data = df, data2 = list(A = A))

#############################################
#modify brms code w/ matrix normal sampling
#############################################

#removed function block w/ Kronecker product function
write(
  "functions {
 /* compute correlated group-level effects with 'by' variables
  * in the presence of a within-group covariance matrix
  * Args:
  *   z: matrix of unscaled group-level effects (M_1 x N_1) or (traits x ids)
  *   SD: matrix of standard deviation parameters (M_1 x Nby_1) or (traits x number of by-factor levels
  *   L: an array of cholesky factor correlation matrices (LCs x number of by-factor levels)
  *   Jby: index which grouping level belongs to which by level (index 1 or 2)
  *   Lcov: cholesky factor of within-group correlation matrix (LC of A)
  * Returns:
  *   matrix of scaled group-level effects
  */
  matrix scale_r_cor_by_cov(matrix z, matrix SD, matrix[] L,
                            int[] Jby, matrix Lcov) {
    matrix[rows(L[1]), cols(L[1])] LC[size(L)];
    matrix[cols(z), rows(z)] r_by[size(LC)];
    matrix[cols(z),rows(z)] r;

    for (i in 1:size(LC)) {
      LC[i] = diag_pre_multiply(SD[, i], L[i]);
      r_by[i] = Lcov * z' * diag_matrix(rep_vector(1,rows(z)));
    }
    for (j in 1:cols(z)) {
    r[j] = r_by[Jby[j]][j] * LC[Jby[j]];
    }
  return r;
  }
}

data {
  int<lower=1> N;  // total number of observations
  int<lower=1> N_t1;  // number of observations
  vector[N_t1] Y_t1;  // response variable
  int<lower=1> N_t2;  // number of observations
  vector[N_t2] Y_t2;  // response variable
  int<lower=1> nresp;  // number of responses
  int nrescor;  // number of residual correlations
  // data for group-level effects of ID 1
  int<lower=1> N_1;  // number of grouping levels
  int<lower=1> M_1;  // number of coefficients per level
  int<lower=1> J_1_t1[N_t1];  // grouping indicator per observation
  int<lower=1> J_1_t2[N_t2];  // grouping indicator per observation
  int<lower=1> Nby_1;  // number of by-factor levels
  int<lower=1> Jby_1[N_1];  // by-factor indicator per observation
  matrix[N_1, N_1] Lcov_1;  // cholesky factor of known covariance matrix
  // group-level predictor values
  vector[N_t1] Z_1_t1_1;
  vector[N_t2] Z_1_t2_2;
  int<lower=1> NC_1;  // number of group-level correlations
  int prior_only;  // should the likelihood be ignored?
}
transformed data {
  vector[nresp] Y[N];  // response array
  for (n in 1:N) {
    Y[n] = transpose([Y_t1[n], Y_t2[n]]);
  }
}
parameters {
  real<lower=0> sigma_t1;  // dispersion parameter
  real<lower=0> sigma_t2;  // dispersion parameter
  cholesky_factor_corr[nresp] Lrescor;  // parameters for multivariate linear models
  matrix<lower=0>[M_1, Nby_1] sd_1;  // group-level standard deviations
  matrix[M_1, N_1] z_1;  // standardized group-level effects
  cholesky_factor_corr[M_1] L_1[Nby_1];  // cholesky factor of correlation matrix
}
transformed parameters {
  matrix[N_1, M_1] r_1;  // actual group-level effects
  // using vectors speeds up indexing in loops
  vector[N_1] r_1_t1_1;
  vector[N_1] r_1_t2_2;
  // compute actual group-level effects
  r_1 = scale_r_cor_by_cov(z_1, sd_1, L_1, Jby_1, Lcov_1);
  r_1_t1_1 = r_1[, 1];
  r_1_t2_2 = r_1[, 2];
}
model {
  // likelihood including constants
  if (!prior_only) {
    // initialize linear predictor term
    vector[N_t1] mu_t1 = rep_vector(0.0, N_t1);
    // initialize linear predictor term
    vector[N_t2] mu_t2 = rep_vector(0.0, N_t2);
    // multivariate predictor array
    vector[nresp] Mu[N];
    vector[nresp] sigma = transpose([sigma_t1, sigma_t2]);
    // cholesky factor of residual covariance matrix
    matrix[nresp, nresp] LSigma = diag_pre_multiply(sigma, Lrescor);
    for (n in 1:N_t1) {
      // add more terms to the linear predictor
      mu_t1[n] += r_1_t1_1[J_1_t1[n]] * Z_1_t1_1[n];
    }
    for (n in 1:N_t2) {
      // add more terms to the linear predictor
      mu_t2[n] += r_1_t2_2[J_1_t2[n]] * Z_1_t2_2[n];
    }
    // combine univariate parameters
    for (n in 1:N) {
      Mu[n] = transpose([mu_t1[n], mu_t2[n]]);
    }
    target += multi_normal_cholesky_lpdf(Y | Mu, LSigma);
  }
  // priors including constants
  target += exponential_lpdf(sigma_t1 | 1);
  target += exponential_lpdf(sigma_t2 | 1);
  target += lkj_corr_cholesky_lpdf(Lrescor | 1);
  target += exponential_lpdf(to_vector(sd_1) | 1);
  target += std_normal_lpdf(to_vector(z_1));
  target += lkj_corr_cholesky_lpdf(L_1[1] | 1);
  target += lkj_corr_cholesky_lpdf(L_1[2] | 1);
}
generated quantities {
  // residual correlations
  corr_matrix[nresp] Rescor = multiply_lower_tri_self_transpose(Lrescor);
  vector<lower=-1,upper=1>[nrescor] rescor;
  // compute group-level correlations
  corr_matrix[M_1] Cor_1_1 = multiply_lower_tri_self_transpose(L_1[1]);
  vector<lower=-1,upper=1>[NC_1] cor_1_1;
  // compute group-level correlations
  corr_matrix[M_1] Cor_1_2 = multiply_lower_tri_self_transpose(L_1[2]);
  vector<lower=-1,upper=1>[NC_1] cor_1_2;
  // extract upper diagonal of correlation matrix
  for (k in 1:nresp) {
    for (j in 1:(k - 1)) {
      rescor[choose(k - 1, 2) + j] = Rescor[j, k];
    }
  }
  // extract upper diagonal of correlation matrix
  for (k in 1:M_1) {
    for (j in 1:(k - 1)) {
      cor_1_1[choose(k - 1, 2) + j] = Cor_1_1[j, k];
    }
  }
  // extract upper diagonal of correlation matrix
  for (k in 1:M_1) {
    for (j in 1:(k - 1)) {
      cor_1_2[choose(k - 1, 2) + j] = Cor_1_2[j, k];
    }
  }
}
", "m_by.stan")

#####################################################################
#compare times and estimated phylogenetic correlations
#####################################################################

m1 = stan_model("m_1.stan")
m2 = stan_model("m_by.stan")

#compare bias of approaches
run = 200 #number of random samples
med_brm = data.frame(n = seq(1:run), time = NA, cor1_emp_bias = NA, cor2_emp_bias = NA)
med_mtn = data.frame(n = seq(1:run), time = NA, cor1_emp_bias = NA, cor2_emp_bias = NA)


counter = 0 #track progress
for(i in 1:run){
#######################
#sim data
#######################

#simulate scaling matrix
A = rethinking::rlkjcorr(1, 100, 1)

#GROUP A
#simulate correlated phylogenetic effects
r_G1 = -0.5 #phylo correlation
v_G =  0.5 #phylo variance

G_cor1 <- matrix(c(1,r_G1,r_G1,1), nrow=2, ncol=2)
G_sd1  <- c(sqrt(v_G),sqrt(v_G))
G1 <- diag(G_sd1) %*% G_cor1 %*% diag(G_sd1)

r_G2 =  0.5 #phylo correlation
v_G =  0.5 #phylo variance

G_cor2 <- matrix(c(1,r_G2,r_G2,1), nrow=2, ncol=2)
G_sd2  <- c(sqrt(v_G),sqrt(v_G))
G2 <- diag(G_sd2) %*% G_cor2 %*% diag(G_sd2)

Gblock = as.matrix(Matrix::bdiag(G1,G2))

Kron.prod1 <- Gblock %x% A
P <- matrix(mvtnorm::rmvnorm(1, mean=rep(0,nrow(A)*4), sigma=Kron.prod1), ncol = 4)
P1 = P[c(1:50),1:2]
emp_cor1 = cor(P1)[2,1]
```

```r
P2 = P[c(51:100),3:4]
emp_cor2 = cor(P2)[2,1]

#group id
P = rbind(P1,P2)

#Gaussian responses
v_res = 0.5 #residual variance (assume independent errors)
t1 = 0 + P[,1] + rnorm(nrow(A), 0, sqrt(v_res))
t2 = 0 + P[,2] + rnorm(nrow(A), 0, sqrt(v_res))

library(brms); library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())

df = data.frame(t1,t2, phylo = seq(1:nrow(A)),
                group = rep(c(1, 2), each = nrow(A)/2))

rownames(A) = seq(1:nrow(A))

#get stan data list
stan_data = make_standata(formula = bf(t1 ~ 0 + (1|G| gr(phylo, cov = A, by = group))) +
                                     bf(t2 ~ 0 + (1|G| gr(phylo, cov = A, by = group)) ),
                            data = df, data2 = list(A = A))

#current brms
#####################################################################
start_time <- Sys.time() #time model
mod1 <- sampling(m1, data= stan_data, init = 0, iter = 2000, warmup = 1000)
med_brm[i,"time"] = Sys.time() - start_time

post1 = extract(mod1)
med_brm[i,"cor1_emp_bias"] = median(post1$cor_1_1) - emp_cor1
med_brm[i,"cor2_emp_bias"] = median(post1$cor_1_2) - emp_cor2

#updated code
#####################################################################
start_time = Sys.time()
mod2 <- sampling(m2, data = stan_data, init = 0, iter = 2000, warmup = 1000)
med_mtn[i,"time"] = Sys.time() - start_time

post2 = extract(mod2)
med_mtn[i,"cor1_emp_bias"] = median(post2$cor_1_1) - emp_cor1
med_mtn[i,"cor2_emp_bias"] = median(post2$cor_1_2) - emp_cor2


saveRDS(med_brm,"med_brm_group.RDS")
saveRDS(med_mtn,"med_mtn_group.RDS")

counter <- counter + 1;
print(paste(counter/run*100, "% has been processed"))

}
```

```r
med_brm$type = "current brms"
med_mtn$type = "matrix normal"
ldf = rbind(med_brm, med_mtn)


######################################################################
#plot comparisons of group-specific phylogenetic correlations
######################################################################

#compare computational time
ldf$min = ifelse(ldf$time>10, ldf$time/60, ldf$time)  #change seconds to minutes

p1 =
  ggplot(ldf, aes(min, fill = type)) +
  geom_histogram(aes(y=..count../run), color = "black", position='identity',
                 binwidth=0.20, alpha = 0.20)+
  scale_fill_manual(values = c("red", "blue"), name = "Method")+
  geom_vline(xintercept=0)+
  ggtitle("Computation time")+
  xlab("Minutes (binwidth = 0.20)")+
  ylab("Proportion of 200 samples\n")+
  theme(panel.background = element_rect(fill='white', colour='black'),
        axis.title = element_text(size = 12))

#sample bias 1
p2 =
  ggplot(ldf, aes(cor1_emp_bias, fill = type)) +
  geom_histogram(aes(y=..count../run), color = "black", position='identity',
                 binwidth=0.05, alpha = 0.20)+
  scale_fill_manual(values = c("red", "blue"), name = "Method")+
  geom_vline(xintercept=0)+
  ggtitle("Bias in sample phylogenetic correlation (Group #1)")+
  xlab("Posterior median r - sample r (binwidth = 0.05)")+
  ylab("Proportion of 200 samples\n")+
  theme(panel.background = element_rect(fill='white', colour='black'),
        axis.title = element_text(size = 12))

#sample bias 2
p3 =
  ggplot(ldf, aes(cor2_emp_bias, fill = type)) +
  geom_histogram(aes(y=..count../run), color = "black", position='identity',
                 binwidth=0.05, alpha = 0.20)+
  scale_fill_manual(values = c("red", "blue"), name = "Method")+
  geom_vline(xintercept=0)+
  ggtitle("Bias in sample phylogenetic correlation (Group #2)")+
  xlab("Posterior median r - sample r (binwidth = 0.05)")+
  ylab("Proportion of 200 samples\n")+
  theme(panel.background = element_rect(fill='white', colour='black'),
        axis.title = element_text(size = 12))


library(cowplot)
```

```r
p = plot_grid(p1, p2,  p3, ncol = 3)
save_plot(p, filename= "matrix normal brms comparison.png", base_height = 4, base_width = 15)
```